Department of Computer Science and Engineering

Python Programming(ITO- 804): Assignment 1

Name: Sushant Slathia

Roll No: 191203037

Semester: 8th A1

Submitted to: Mr. Saurabh Sharma

Index

S.no	Question	Marks
1	Write a Python program that takes a list of daily stock prices as input, and returns the best days to buy and sell stocks in order to maximize profit. The list contains the stock prices for each day, starting from the first day. For example, the list (100, 180, 260, 310, 40, 535, 695) represents the stock prices for 7 days, where the price on the first day is 100, the second day is 180, and so on. The program should find the best days to buy and sell stocks such that the profit obtained is maximum. For instance, in the given list of stock prices, the best days to buy and sell stocks would be: Buy stock on the 1st day (price=100) Sell stock on the 4th day (price=310) Buy stock on the 5th day (price=40) Sell stock on the 7th day (price=695) The program should output these buy and sell days as a tuple or list of integers.	2.5
2	You are given a list of book titles and their corresponding publication years. Your task is to find the earliest year in which a trilogy of books was published. A trilogy is defined as a series of three books published in consecutive years. For example, consider the following list of book titles and publication years: titles = ['The Hunger Games', 'Catching Fire', 'Mockingjay', 'The Lord of the Rings', 'The Two Towers', 'The Return of the King', 'Divergent', 'Insurgent', 'Allegiant'] years = [2008, 2009, 2010, 1954, 1955, 1956, 2011, 2012, 2013] The earliest year in which a trilogy was published is 1954. Write a Python function earliest_trilogy_year(titles: List[str], years: List[int]) -> Optional[int] that takes two lists as input: titles containing the titles of the books, and years containing their corresponding publication years. The function should return the earliest year in which a trilogy of books was published, or None if no such trilogy exists. Examples: titles = ['Book1', 'Book2', 'Book3', 'Book4', 'Book5', 'Book6'] years = [2019, 2021, 2012, 2013, 2016, 2017] print(earliest_trilogy_year(titles, years)) The earliest year in which a trilogy was published is : None A trilogy is defined as a series of three books published in consecutive years. Note: • You can assume that the input lists are non-empty and contain an equal number of elements. • If multiple trilogies exist with the same earliest year, return that year.	2.5
3	Write a Python program that reads in a CSV file of stock prices (e.g. ticker symbol, date, price), and then uses dictionaries and lists to calculate the highest and lowest prices for each stock from following table.	2.5
4	A) Write a Python program to remove duplicates from a list of lists. Sample list: [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]] B) Write a Python program which takes a list and returns a list with the elements "shifted left by one position" so [1, 2, 3] yields [2, 3, 1]. Example: [1, 2, 3] → [2, 3, 1] [11, 12, 13] → [12, 13, 11] C) Iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element. Original list [11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89] And expected output is: Printing count of each item {11: 3, 45: 3, 8: 1, 23: 2, 89: 2}	2.5

Q1. Write a Python program that takes a list of daily stock prices as input, and returns the best days to buy and sell stocks in order to maximize profit. The list contains the stock prices for each day, starting from the first day. For example, the list (100, 180, 260, 310, 40, 535, 695) represents the stock prices for 7 days, where the price on the first day is 100, the second day is 180, and so on. The program should find the best days to buy and sell stocks such that the profit obtained is maximum. For instance, in the given list of stock prices, the best days to buy and sell stocks would be: Buy stock on the 1st day (price=100) Sell stock on the 4th day (price=310) Buy stock on the 5th day (price=40) Sell stock on the 7th day (price=695) The program should output these buy and sell days as a tuple or list of integers.

```
def find_best_days(prices):
    """
    Given a list of daily stock prices, returns the best days to buy and sell stocks in order to maximize profit.
    :param prices: a list of integers representing the stock prices for each day, starting from the first day
    :return: a tuple of integers representing the best days to buy and sell stocks
    """
    n = len(prices)
    if n < 2:
        return ()

    buy_day, sell_day = 0, 0
    max_profit = 0

    for i in range(1, n):
        if prices[i] < prices[buy_day]:
            buy_day = i
        elif prices[i] - prices[buy_day] > max_profit:
            max_profit = prices[i] - prices[buy_day]
            sell_day = i

    if max_profit == 0:
        return ()
    else:
        return (buy_day+1, sell_day+1)
```

To use the function, you can simply call it with a list of daily stock prices as argument:

```
prices = [100, 180, 260, 310, 40, 535, 695]

best_days = find_best_days(prices)

print(best_days)
```

The output is a tuple of integers representing the best days to buy and sell stocks. In this example, the best days are (1, 4) and (5, 7), which means you should buy stock on the 1st day and sell it on the 4th day, and then buy stock on the 5th day and sell it on the 7th day, in order to maximize profit.

Q2. You are given a list of book titles and their corresponding publication years. Your task is to find the earliest year in which a trilogy of books was published. A trilogy is defined as a series of three books published in consecutive years. For example, consider the following list of book titles and publication years: titles = ['The Hunger Games', 'Catching Fire', 'Mockingjay', 'The Lord of the Rings', 'The Two Towers', 'The Return of the King', 'Divergent', 'Insurgent', 'Allegiant'] years = [2008, 2009, 2010, 1954, 1955, 1956, 2011, 2012, 2013] The earliest year in which a trilogy was published is 1954. Write a Python function earliest_trilogy_year(titles: List[str], years: List[int]) -> Optional[int] that takes two lists as input: titles containing the titles of the books, and years containing their corresponding publication years. The function should return the earliest year in which a trilogy of books was published, or None if no such trilogy exists. Examples: titles = ['Book1', 'Book2', 'Book3', 'Book4', 'Book5', 'Book6'] years = [2019, 2021, 2012, 2013, 2016, 2017] print(earliest_trilogy_year(titles, years)) The earliest year in which a trilogy was published is : None A trilogy is defined as a series of three books published in consecutive years. Note: • You can assume that the input lists are non-empty and contain an equal number of elements. • If multiple trilogies exist with the same earliest year, return that year.

```python
from typing import List, Optional

def earliest_trilogy_year(titles: List[str], years: List[int]) -> Optional[int]:
    # sort the books by their publication years
    sorted_books = sorted(zip(years, titles))
    n = len(sorted_books)
    for i in range(n - 2):
        # check if the current book and the next two books were published in consecutive years
        if sorted_books[i][0] + 1 == sorted_books[i + 1][0] and sorted_books[i + 1][0] + 1 == sorted_books[i + 2][0]:
            return sorted_books[i][0]
    return None
```

Function Working: -

1. First, the books are sorted by their publication years using the 'zip()'function to pair each book's title with its corresponding year.
2. Then, the function iterates through the sorted list of books using a loop that runs from 0 to n-3, where n is the length of the list. This ensures that there are at least two more books after the current book in the loop.
3. Inside the loop, the function checks if the current book and the next two books were published in consecutive years. If so, the function returns the year of the first book in the trilogy.
4. If no trilogy is found in the loop, the function returns None.

For example, if we call earliest_trilogy_year() with the following inputs:

```python
titles = ['The Hunger Games', 'Catching Fire', 'Mockingjay', 'The Lord of the Rings', 'The Two Towers', 'The Return of the King',
years = [2008, 2009, 2010, 1954, 1955, 1956, 2011, 2012, 2013]
print(earliest_trilogy_year(titles, years))
```

This is because the earliest trilogy in the list is "The Lord of the Rings", which was published in 1954, 1955, and 1956.

Q3. WRITE A PYTHON PROGRAM THAT READS IN A CSV FILE OF STOCK PRICES (E.G. TICKER SYMBOL, DATE, PRICE), AND THEN USES DICTIONARIES AND LISTS TO CALCULATE THE HIGHEST AND LOWEST PRICES FOR EACH STOCK FROM FOLLOWING TABLE:

Symbol	Date	Price
AAPL	2022-01-01	135.90
AAPL	2022-01-02	138.45
AAPL	2022-01-03	142.20
GOOG	2022-01-01	2105.75
GOOG	2022-01-02	2098.00
GOOG	2022-01-03	2125.50
MSFT	2022-01-01	345.20
MSFT	2022-01-02	344.70
MSFT	2022-01-03	342.10

```
import csv
with open('3_csv.csv') as file:
reader = csv.reader(file)
next(reader) # Skip the header row
prices = {} # Create an empty dictionary to store the prices for each stock
for row in reader:
# Extract the symbol, date, and price from the row
ticker, date, price = row
# Convert the price from a string to a float
price = float(price)
# Check if the ticker symbol is already in the dictionary
if ticker in prices:
prices[ticker].append(price)
else:
prices[ticker] = [price]
for ticker, price_list in prices.items():
highest_price = max(price_list)
lowest_price = min(price_list)
print(f"{ticker}: Highest
Price = ${highest_price:.2f},Lowest Price = ${lowest_price:.2f}")
```

Q4. (A) WRITE A PYTHON PROGRAM TO REMOVE DUPLICATES FROM A LIST OF LISTS.

SAMPLE LIST: [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]

```
original_list = [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]

new_list = [list(x) for x in set(tuple(x) for x in original_list)]

print(new_list)
```

HERE WE HAVE USED THE 'SET' BUILT-IN-FUNCTION.

OUTPUT: -

```
[[40], [33], [30, 56, 25], [10, 20]]


...Program finished with exit code 0
Press ENTER to exit console.
```

B) WRITE A PYTHON PROGRAM WHICH TAKES A LIST AND RETURNS A LIST WITH THE ELEMENTS "SHIFTED LEFT BY ONE POSITION" SO [1, 2, 3] YIELDS [2, 3, 1].

EXAMPLE: [1, 2, 3] → [2, 3, 1] [11, 12, 13] → [12, 13, 11]

```python
def shift_left(lst):
    if len(lst) <= 1:
        return lst
    else:
        return lst[1:] + [lst[0]]

# example usage
lst = [1, 2, 3]
shifted_lst = shift_left(lst)
print(shifted_lst)
```

THIS PROGRAM DEFINES A FUNCTION CALLED 'SHIFT_LEFT' THAT TAKES A LIST 'LST' AS INPUT AND RETURNS A NEW LIST WITH THE ELEMENTS SHIFTED LEFT BY ONE POSITION. THE FUNCTION FIRST CHECKS IF THE LENGTH OF THE LIST IS LESS THAN OR EQUAL TO 1, IN WHICH CASE IT JUST RETURNS THE ORIGINAL LIST. OTHERWISE, IT RETURNS A NEW LIST CREATED BY CONCATENATING THE SLICE OF THE LIST FROM INDEX 1 TO THE END WITH A LIST CONTAINING JUST THE FIRST ELEMENT OF THE ORIGINAL LIST.

C) ITERATE A GIVEN LIST AND COUNT THE OCCURRENCE OF EACH ELEMENT AND CREATE A DICTIONARY TO SHOW THE COUNT OF EACH ELEMENT. ORIGINAL LIST [11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89] AND EXPECTED OUTPUT IS: PRINTING COUNT OF EACH ITEM {11: 3, 45: 3, 8: 1, 23: 2, 89: 2}

```python
original_list = [11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89]

count_dict = {}
for element in original_list:
    if element in count_dict:
        count_dict[element] += 1
    else:
        count_dict[element] = 1

print("Printing count of each item", count_dict)
```

OUTPUT: -

```
Printing count of each item {11: 3, 45: 3, 8: 1, 23: 2, 89: 2}
```

EXPLANATION:

1. THE ORIGINAL LIST IS DEFINED.
2. A DICTIONARY CALLED 'COUNT_DICT' IS DEFINED TO STORE THE COUNT OF EACH ELEMENT.
3. A FOR LOOP IS USED TO ITERATE THROUGH EACH ELEMENT IN THE ORIGINAL LIST.
4. IF THE ELEMENT IS ALREADY IN THE 'COUNT_DICT', ITS COUNT IS INCREMENTED BY 1.
5. IF THE ELEMENT IS NOT IN THE 'COUNT_DICT', A NEW KEY-VALUE PAIR IS ADDED WITH THE KEY BEING THE ELEMENT AND THE VALUE BEING 1.
6. FINALLY, THE 'COUNT_DICT' IS PRINTED TO SHOW THE COUNT OF EACH ELEMENT.